CERTIFICATE OF TRANSMISSION

I hereby certify that this correspondence (along with any paper referred to as being attached or enclosed) is being submitted *via* the USPTO EFS Filing System on the date shown below to **Mail Stop Appeal Brief-Patents**, Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

Date: September 18, 2008 /Jessica Sexton/
Jessica Sexton

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re patent application of:			
Applicant(s):	Sean E. Trowbridge	Examiner:	James D. Rutten
Serial No:	09/817,880	Art Unit:	2192
Filing Date:	March 26, 2001		

Title: SYSTEM AND METHOD FOR PROVIDING ON-DEMAND GENERATION OF SPECIALIZED EXECUTABLES

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

REPLY TO EXAMINER'S ANSWER DATED JULY 18, 2008

Dear Sir:

Appellants' representative submits this reply brief to the Examiner's Answer. Claim 33 has been correct throughout the brief to match the entered amendments to claim 33. In addition, the grounds of rejection section VI, parts A and B have been corrected to reflect the claims being argued under those parts in section VII.

I. Real Party in Interest (37 C.F.R. §41.37(c)(1)(i))

The real party in interest in the present appeal is Microsoft Corporation, the assignee of the present application.

II. Related Appeals and Interferences (37 C.F.R. §41.37(c)(1)(ii))

Appellants, appellants' legal representative, and/or the assignee of the present application are not aware of any appeals or interferences which may be related to, will directly affect, or be directly affected by or have a bearing on the Board's decision in the pending appeal.

III. Status of Claims (37 C.F.R. §41.37(c)(1)(iii))

Claims 1-33 stand rejected by the Examiner. The rejection of claims 1-33 is being appealed.

IV. Status of Amendments (37 C.F.R. §41.37(c)(1)(iv))

Claims 30 and 33 were amended after the Final Office Action dated October 16, 2006 to address issues related to the rejection under 35 U.S.C. §101. The claim amendments were entered by the Examiner.

V. Summary of Claimed Subject Matter (37 C.F.R. §41.37(c)(1)(v))

A. Independent Claim 1

Independent claim 1 recites a computer implemented system for generating specialized executables, comprising the following computer executable components:

a virtual subsystem that processes a generic code image; (*See e.g.*, page 5, lines 20-24)

a loader to determine availability of a specialized image that is associated with an operating environment of the virtual subsystem; and (*See e.g.*, page 3, lines 15-24; page 6, lines 8-12)

a log to store generic code image runtime information relating to the operating environment of the virtual subsystem, the runtime logged information includes at least a set of information related to a particular user to create a native executable according to the particular user, the logged information is employed as feedback to generate the native executable based

upon the availability of the specialized image, the native executable is utilized to provide improved performance of the virtual subsystem. (*See e.g.*, page 6, lines 15-22)

B. Independent Claim 20

Independent claim 20 recites A computer implemented method to generate runtime code, comprising the following computer executable acts:

determining a first code image associated with a possible runtime environment; (*See e.g.*, page 10, lines 25, page 11, line 3)

executing the first code image in an unmodified form in the runtime environment; (*See e.g.*, page 5, lines 7-9, page 11, lines 3-5)

and

generating runtime feedback associated with the first code image and a particular user to adjust a subsequent code image according to the runtime environment, the feedback includes at least a set of information to create a code image according to the particular user. (*See e.g.*, page 6, lines 15-22; page 11, lines 5-14)

C. Independent Claim 28

Independent claim 28 recites a system to generate runtime code, comprising:

means for determining a specialized code image associated with a possible runtime environment; (*See e.g.*, page 3, lines 15-24; page 6, lines 8-12; page 10, lines 25, page 11, line 3)

means for executing the specialized code image in an unmodified form in the runtime environment; and (*See e.g.*, page 5, lines 7-23, page 6, lines 25-28; page 11, lines 3-5)

means for generating runtime feedback associated with the specialized code image and a particular user to adjust a subsequent specialized code image according to the runtime environment, the feedback includes at least a set of information to create a specialized code image according to the particular user. (*See e.g.*, page 6, lines 15-22; page 7, line 2- page8, line 14; page 11, lines 5-14)

D. Independent Claim 30

Independent claim 30 recites a computer implemented signal transmitted between at least two processes executing on one or more computers facilitating generation of specialized executables, comprising:

a signal for communicating between one or more components of a virtual system, the virtual system processing a generic code image and logging information relating to an operating environment and a particular user of the virtual system *via* the signal, the logged information includes at least a set of information to create a specialized native executable according to the particular user; (*See e.g.*, page 6, lines 15-22; page 7, line 2-page 8, line 14; page 11, lines 5-14)

wherein the logged information is employed as feedback across the signal to generate a specialized native executable if the generic code image is determined incompatible with the operating environment of the virtual system, the specialized native executable is utilized to provide improved performance of the virtual system. (*See e.g.*, page 6, lines 15-22)

E. Independent Claim 32

Independent claim 32 recites a computer implemented means having stored thereon a data structure employed to create an executable image, comprising:

a first data field having parameters relating to at least one of an operating system version, a processor type, a virtual system version, and a processor specifier; (*See e.g.*, page 3, lines 25-28; page 5, lines 27-28)

a second data field having at least one of a developer parameter, a domain flag, a security information field, and a binding information field; and (*See e.g.*, page 3, lines 25-28; page 5, lines 27-28)

a third data field having at least a set of information associated with a particular user that is logged during execution of a virtual system to create an executable image according to the particular user of the virtual system. (*See e.g.*, page 6, lines 15-22)

F. Independent Claim 33

Independent claim 33 recites a computer implemented system, comprising the following computer executable components:

an execution engine that processes an Intermediate Language (IL) image, the execution engine generating operating environment data associated with a particular user while processing the IL image; (*See e.g.*, page 6, lines 15-page 8, line 1)

a specialized executable image generated at least in part from the operating environment data, the operating environment data includes at least a set of information to create a specialized executable image according to the particular user, the specialized executable image stored in a repository of one or more other specialized executable images; and (*See e.g.*, page 6, lines 15-page 8, line 1)

wherein the execution engine selects at least one specialized executable image from the repository if the at least one specialized image matches present operating environment data. (*See e.g.*, page 6, lines 15-page 8, line 1)

VI. Grounds of Rejection to be Reviewed (37 C.F.R. §41.37I(1)(vi))

A. Whether claims 1, 2, 5-17 and 19 are unpatentable under 35 U.S.C. §103(a) over Breslau *et al.* (U.S. 5,761,512), Spyker *et al.* (U.S. 6,571,389), Armstrong ("Hotspot: A new breed of virtual machine"), and Knight (U.S. 6, 126,330).

B. Whether claims 30 and 31 are unpatentable under 35 U.S.C. §103(a) over Breslau *et al.* (U.S. 5,761,512), Spyker *et al.* (U.S. 6,571,389), Armstrong ("Hotspot: A new breed of virtual machine"), and Knight (U.S. 6, 126,330).

C. Whether claims 3 and 4 are unpatentable under 35 U.S.C. §103(a) over Breslau *et al.*, Spyker *et al.*, Armstrong, and Knight, and further in view of Fogarty *et al.* (U.S. 6,721,946).

D. Whether claim 18 is unpatentable under 35 U.S.C. §103(a) over Breslau *et al.*, Spyker *et al.*, Armstrong, and Knight, and further in view of Nelin *et al.* (U.S. 6,253,368).

E. Whether claims 20-22 and 27-29 are unpatentable under 35 U.S.C. §103(a) over Goodwin *et al.* (U.S. 6,158,049) in view of Knight (U.S. 6,126,330).

F. Whether claims 23 and 24 are unpatentable under 35 U.S.C. §103(a) over Goodwin *et al.* and Knight, further in view of Aho *et al.* ("Compilers: Principles, Techniques, and Tools").

G. Whether claims 25 and 26 are unpatentable under 35 U.S.C. §103(a) over Goodwin *et al*., Knight, and Aho *et al*., and further in view of Breslau.

H. Whether claims 32 is unpatentable under 35 U.S.C. §103(a) over Breslau in view of Spyker, Knight, and Nelin.

J. Whether claims 33 is unpatentable under 35 U.S.C. §103(a) over Breslau in view of Ramezani (U.S. 6,457,122), Knight, and Spyker.

VII. Argument (37 C.F.R. §41.37(c)(1)(vii))

A. Rejection of Claims 1, 2, 5-17 and 19 Under 35 U.S.C §103(a)

Claims 1, 2, 5-17 and 19 stand rejected under 35 U.S.C. §103(a) as being unpatentable over prior art of record Breslau *et al.* (U.S. 5,761,512) in view of Spyker *et al.* (U.S. 6,571,389), Armstrong ("Hotspot: A new breed of virtual machine"), and Knight (U.S. 6, 126,330). It is respectfully submitted that this rejection should be reversed for at least the following reasons. Breslau *et al*., Spyker *et al*., Armstrong, and Knight, alone or in combination, do not teach or suggest each and every limitation of applicant's claimed invention.

> To reject claims in an application under §103, an examiner must establish a *prima facie* case of obviousness. A *prima facie* case of obviousness is established by a showing of three basic criteria. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. *See* MPEP §706.02(j). The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art and not based on applicant's disclosure. *See In re Vaeck*, 947 F.2d 488, 20 USPQ2d 1438 (Fed. Cir. 1991).

The subject invention relates to creating and loading an appropriate executable image at run time in a virtual execution environment based upon attributes associated with the operating environment and user of the virtual execution environment. For example, the attributes can

related to a specific user that is running applications in the operating environment, thus allowing for the subject invention to create one or more executable images that are optimized for the user's current needs. Independent claim 1 (and similarly independent claim 30) recites *a log to store generic code image runtime information relating to the operating environment of the virtual subsystem*, ***the runtime logged information includes at least a set of information related to a particular user to create a native executable according to the particular user, the logged information is employed as feedback to generate the native executable based upon the availability of the specialized image***.

Breslau *et al*., Spyker *et al*., and Armstrong fail to teach or suggest the aforementioned novel aspects of applicant's invention as recited in the subject claims. The Office Action asserts that Breslau *et al*. discloses logged information to create a native executable. However, the information disclosed in the cited art is a table that equates affinity values to runtime environments. This is not information that is logged during runtime execution of an executable image. The cited art does not disclose or suggest how this table is populated and specifically is silent regarding logging information during runtime execution of the compiled image as in applicant's claimed invention. Breslau *et al*. is concerned with creating a single set of code files that contains compile switches (affinity values) that will compile particular code classes for each specific hardware and operating system environment to create an executable image appropriate to the operating environment. Breslau is not concerned with creating a runtime image for a particular user. The Office Action incorrectly asserts that Spyker *et al*. discloses that a user can control the creation of a runtime image according to a particular user. On the contrary, the cited art does not disclose a user specific executable image. Rather, Spyker *et al*. discloses a system for instantiating an appropriate runtime environment for a particular Java application (or applet). Java applications and applets may rely on a particular version of a Java virtual machine (JVM) to execute. Many times the Java virtual machine on particular operating system or associated with a particular browser may be behind or ahead of the version required to execute the Java application. The cited art addresses this disparity by including a properties file in the Java archive file (JAR) associated with a particular Java application. The properties file identifies the particular JVM and other extensions required to execute the Java application. Using this file, the appropriate environment can be instantiated to execute the Java application. The cited art is silent regarding logging runtime information associated with a particular user or creating an

executable image according to the particular user. The section of Spyker *et al.* cited in the Office Action merely states that when a user attempts to launch an application on a computer, the appropriate environment *for the application* is instantiated, *not* an appropriate environment *for a particular user*. It also does not teach or suggest that information related to the user is logged during execution of the application. Moreover, Armstrong is also silent regarding logging runtime information related to a particular user to create an executable image according to the particular user. Armstrong discloses a Java just-in-time compiler to improve execution performance of Java applications. The cited art improves on Java bytecode, by monitoring execution times for methods and compiling those byetcode methods that are running slow into native machine code. No information related to a user is logged. Breslau *et al.*, Spyker *et al.*, and Armstrong do not suggest logging information for particular user or creating a native executable according to a particular user. Knight is cited to make up for these deficiencies of Breslau *et al.*, Spyker *et al.*, and Armstrong. However, Knight discloses a system where a developer creates an input file that is read in by an application to determine instrumentation points for the application. The developer manually enters the instrumentation point information. Knight makes no suggestion that the input file contains runtime logged feedback from the user. Furthermore, this input file is not used to create a native executable image according to a particular user. The input file does not change the application, but merely identifies instrumentation points to monitor. Therefore, Breslau *et al.*, Spyker *et al.*, Armstrong, Knight fail to teach or suggest a log to store generic code image runtime information relating to the operating environment of the virtual subsystem, the runtime logged information includes at least a set of information related to a particular user to create a native executable according to the particular user, the logged information is employed as feedback to generate the native executable based upon the availability of the specialized image as in applicant's claimed invention.

Accordingly, applicant's representative respectfully submits that Breslau *et al.*, Spyker *et al.*, Armstrong, and Knight, alone or in combination, fail to teach or suggest all limitations of applicant's invention as recited in independent claims 1 and 30 (and claims 2, 5-17, 19 and 31 that respectfully depend there from), and thus fails to make obvious the claimed invention. For that reason, reversal of this rejection is respectfully requested.

B. Rejection of Claims 30 and 31 Under 35 U.S.C §103(a)

Claims 30 and 31 stand rejected under 35 U.S.C. §103(a) as being unpatentable over prior art of record Breslau *et al.* (U.S. 5,761,512) in view of Spyker *et al.* (U.S. 6,571,389), Armstrong ("Hotspot: A new breed of virtual machine"), and Knight (U.S. 6, 126,330). It is respectfully submitted that this rejection should be reversed for at least the following reasons. Breslau *et al*., Spyker *et al*., Armstrong, and Knight, alone or in combination, do not teach or suggest each and every limitation of applicant's claimed invention. Independent claim 30 recites *a signal for communicating between one or more components of a virtual system, the virtual system processing a generic code image and* ***logging information relating to*** *an operating environment* ***and a particular user*** *of the virtual system via the signal,* ***the logged information includes at least a set of information to create a specialized native executable according to the particular user****; wherein the logged information is employed as feedback across the signal to generate a specialized native executable if the generic code image is determined incompatible with the operating environment of the virtual system, the specialized native executable is utilized to provide improved performance of the virtual system.* As discusses supra, with respect to the similar limitations of independent claim 1, Breslau *et al*., Spyker *et al*., Armstrong, Knight fail to teach or suggest logging information relating to an operating environment and a particular user of the virtual system via the signal, the logged information includes at least a set of information to create a specialized native executable according to the particular user.

Accordingly, applicant's representative respectfully submits that Breslau *et al*., Spyker *et al*., Armstrong, and Knight, alone or in combination, fail to teach or suggest all limitations of applicant's invention as recited in independent claim 30 (and claim 31 that respectfully depend there from), and thus fails to make obvious the claimed invention. Therefore, reversal of this rejection is respectfully requested.

C. Rejection of Claims 3 and 4 Under 35 U.S.C §103(a)

Claims 3 and 4 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Breslau *et al*., Spyker *et al*., Armstrong, and Knight as applied in the above rejection of claims 1, 2, 5-10, 12-17, 19, 30 and 31, and further in view of Fogarty *et al.* (U.S. 6,721,946). It is respectfully submitted that this rejection should be reversed for at least the following reasons. Fogarty *et al*. does not make up for the aforementioned deficiencies noted above with respect to

Breslau *et al*., Spyker *et al*., Armstrong, and Knight regarding independent claim 1, from which the subject claims depend. The cited reference is related to manufacturing of build to order computers and assessing options configurations that are frequently purchased to produce static installation images. Notably, Fogarty *et al*. fails to teach or suggest a system that contains *a log to store generic code image runtime information relating to the operating environment of the virtual subsystem*, ***the runtime logged information includes at least a set of information related to a particular user to create a native executable according to the particular user, the logged information is employed as feedback to generate the native executable based upon the availability of the specialized image*** as recited in claim 1. Therefore, Breslau *et al*., Spyker *et al*., Armstrong, Knight and Fogarty *et al*. fail to make obvious the subject claimed invention and it is respectfully submitted that this rejection be reversed.

D. Rejection of Claim 18 Under 35 U.S.C §103(a)

Claim 18 stands rejected under 35 U.S.C. §103(a) as being unpatentable over Breslau *et al*., Spyker *et al*., Armstrong, and Knight as applied in the above rejection of claims 1, 2, 5-10, 12-17, 19, 30 and 31, and further in view of Nelin *et al.* (U.S. 6,253,368). It is respectfully submitted that this rejection should be reversed for at least the following reasons. Nelin *et al.* does not make up for the above noted deficiencies with respect to Breslau *et al*., Spyker *et al*., Armstrong, and Knight regarding amended independent claim 1, from which the subject claim depends. Nelin *et al*. discloses a system related to dynamically debugging internet applications. Nelin *et al*. fails to teach or suggest a system that contains *a log to store generic code image runtime information relating to the operating environment of the virtual subsystem*, ***the runtime logged information includes at least a set of information related to a particular user to create a native executable according to the particular user, the logged information is employed as feedback to generate the native executable based upon the availability of the specialized image*** as recited in claim 1. Therefore, Breslau *et al*., Spyker *et al*., Armstrong, Knight and Nelin *et al.* fail to make obvious the subject claimed invention; and it is respectfully submitted that this rejection should be reversed.

E. Rejection of Claims 20-22 and 27-29 Under 35 U.S.C §103(a)

Claims 20-22 and 27-29 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Goodwin *et al.* (U.S. 6,158,049) in view of Knight (U.S. 6,126,330). It is respectfully submitted that this rejection should be reversed for at least the following reasons. Goodwin *et al.* and Knight do not teach each and every element of applicant's invention as recited in the subject claims.

Independent claim 20 (and similarly independent claim 28) recites ***generating runtime feedback associated with the first code image and a particular user to adjust a subsequent code image according to the runtime environment, the feedback includes at least a set of information to create a code image according to the particular user***. Goodwin *et al.* does not teach or suggest the aforementioned novel features of applicant's claimed invention as recited in the subject claims. Rather, Goodwin *et al*. discloses a system whereby code is inserted into an original image as "instrumentation" code and the original image plus the instrumentation code is run to determine a profile that can later be used to generate an optimized version of the original code. The cited art is concerned with automating the process of optimizing native machine code that is generated from high-level language compilers. Goodwin *et al.* is silent regarding any code image runtime feedback related to a particular user and also does not create a native executable image according to a particular user. Knight is cited to make up for these deficiencies of Goodwin *et al*. However, Knight discloses a system where a developer creates an input file that is read in by an application to determine instrumentation points for the application. This input file does not contain runtime logged feedback from the user and Knight does teach or suggest this feature. Furthermore, this input file is not used to create a code image according to a particular user. The input file does not change the application, but merely identifies instrumentation points to monitor. Therefore, Goodwin *et al*. and Knight, alone or in combination, fail to teach or suggest generating runtime feedback associated with the first code image and a particular user to adjust a subsequent code image according to the runtime environment, the feedback includes at least a set of information to create a code image according to the particular user.

In view of at least the foregoing, applicant's representative respectfully submits that Goodwin *et al.* and Knight, alone or in combination, fail to teach or suggest all limitations of applicant's invention as recited in independent claims 20 and 28 (and claims 21, 22, 27 and 29 that

respectfully depend there from), and thus fails to make obvious the subject claimed invention. Accordingly, reversal of this rejection is respectfully requested.

F. Rejection of Claims 23 and 24 Under 35 U.S.C §103(a)

Claims 23 and 24 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Goodwin *et al.* and Knight as applied in the above rejection of claim 21, further in view of Aho *et al.* ("Compilers: Principles, Techniques, and Tools"). It is respectfully requested that this rejection be reversed for at least the following reasons. Aho *et al.* fails to make up for the deficiencies of Goodwin *et al.* and Knight as discussed *supra* with regards to the limitations recited in independent claim 20, from which the subject claims depend. Aho *et al.* describes general principles regarding interpreters, compilers and assembler. The cited art is silent regarding logging runtime information related to a particular user to create an executable image according to the particular user. Therefore, Goodwin *et al.*, Knight and Aho *et al.* fail to teach or suggest ***generating runtime feedback associated with the first code image and a particular user to adjust a subsequent code image according to the runtime environment, the feedback includes at least a set of information to create a code image according to the particular user.*** Therefore, reversal of this rejection is respectfully requested.

G. Rejection of Claims 25 and 26 Under 35 U.S.C §103(a)

Claims 25 and 26 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Goodwin *et al.*, Knight, and Aho *et al.* as applied to the rejection of claim 23 above, and further in view of Breslau. As noted above with respect to independent claim 21 and claim 23 from which claims 25 and 26 depend, Goodwin *et al.*, Knight, and Aho *et al.* fail to teach or suggest ***generating runtime feedback associated with the first code image and a particular user to adjust a subsequent code image according to the runtime environment, the feedback includes at least a set of information to create a code image according to the particular user.*** Breslau *et al.* fails to make up for the deficiencies of Goodwin *et al.*, Knight, and Aho *et al.* with regards to this novel feature. As conceded by the Examiner, with respect to the similar limitation of independent claim 1, Breslau *et al.* fails to teach or suggest runtime feedback associated with a particular and creating a code image according to the particular user. Accordingly, reversal of this rejection is respectfully requested.

H. Rejection of Claim 32 Under 35 U.S.C §103(a)

Claim 32 stands rejected under 35 U.S.C. §103(a) as being unpatentable over Breslau in view of Spyker, Knight, and Nelin. It is respectfully submitted that this rejection should be reversed for at least the following reasons. Breslau *et al*., Spyker *et al*., Knight and Nelin *et al.* are silent regarding ***a set of information associated with a particular user that is logged during execution of a virtual system to create an executable image according to the particular user of the virtual system***, for the reasons discussed above with respect to the similar limitations of independent clams, 1, 20, 28 and 30. Accordingly, reversal of this rejection is respectfully requested..

J. Rejection of Claim 33 Under 35 U.S.C §103(a)

Claim 33 stands rejected under 35 U.S.C. §103(a) as being unpatentable over Breslau in view of Ramezani (U.S. 6,457,122), Knight, and Spyker. It is respectfully submitted that this rejection should be reversed for at least the following reasons. Breslau *et al*., Ramezani, Knight and Spyker *et al*., alone or in combination, do not teach or suggest each and every limitation of applicant's claimed invention. Ramezani fails to make up for the deficiencies of Breslau *et al*., Knight and Spyker *et al*. as discussed *supra* with regards to the similar limitations recited in independent claim 1. The cited art discloses a system for installing programs on writeable storage device in a fault tolerable manner. Ramezani is silent regarding *a specialized executable image generated at least in part from the operating environment data****, the operating environment data includes at least a set of information to create a specialized executable image according to the particular user*** as recited in independent claim 33. In view of the foregoing, reversal of this rejection is respectfully requested.

CONCLUSION

The present application is believed to be in condition for allowance in view of the above comments. A prompt action to such end is earnestly solicited.

In the event any fees are due in connection with this document, the Commissioner is authorized to charge those fees to Deposit Account No. 50-1063 [MSFTP197US].

Should the Examiner believe a telephone interview would be helpful to expedite favorable prosecution, the Examiner is invited to contact appellants' undersigned representative at the telephone number below.

Respectfully submitted,

AMIN, TUROCY & CALVIN, LLP

/Himanshu S. Amin/
Himanshu S. Amin
Reg. No. 40,894

AMIN, TUROCY & CALVIN, LLP
24TH Floor, National City Center
1900 E. 9TH Street
Cleveland, Ohio 44114
Telephone (216) 696-8730
Facsimile (216) 696-8731

VIII. Claims Appendix (37 C.F.R. §41.37(c)(1)(viii))

1. A computer implemented system for generating specialized executables, comprising the following computer executable components:

a virtual subsystem that processes a generic code image;

a loader to determine availability of a specialized image that is associated with an operating environment of the virtual subsystem; and

a log to store generic code image runtime information relating to the operating environment of the virtual subsystem, the runtime logged information includes at least a set of information related to a particular user to create a native executable according to the particular user, the logged information is employed as feedback to generate the native executable based upon the availability of the specialized image, the native executable is utilized to provide improved performance of the virtual subsystem.

2. The system of claim 1, wherein the native executable is selected for execution by the virtual subsystem by matching a current environment setting with the logged information.

3. The system of claim 1, further comprising an image repository to store 1 through N specialized native images, wherein N is a positive integer.

4. The system of claim 3, wherein the image repository is a local or remote database.

5. The system of claim 1, further comprising at least one of a local or remote data log to store the logged information.

6. The system of claim 5, wherein a data log stores 1 though N environment parameter descriptions associated with 1 to N encountered images, wherein N is a positive integer.

7. The system of claim 1, wherein the virtual subsystem includes at least one of an execution engine and a virtual machine, and wherein the generic code image is an intermediate language image.

8. The system of claim 7, wherein the virtual subsystem employs Just-In-Time compilation on the generic code image.

9. The system of claim 7, wherein the virtual subsystem performs at least one of loads related executables, organizes data fields and methods within the generic image, analyzes intermediate language formats and generates native platform code.

10. The system of claim 7, wherein the virtual subsystem creates the native executable the time the generic code image is installed.

11. The system of claim 1, wherein the logged information includes a set of information to create executable images according to at least one of a method of invocation and a usage pattern.

12. The system of claim 1, wherein a native code image is generated by the virtual subsystem.

13. The system of claim 1, further comprising an image generator for processing the feedback and generating the native executable.

14. The system of claim 13, wherein the image generator further comprises at least one of a compiler and a code processor.

15. The system of claim 14, further comprising an image processing tool to read the logged information and associate one or more environmental settings with one or more related images encountered during virtual subsystem execution.

16. The system of claim 1, wherein the logged information includes parameters relating to at least one of an operating system version, a processor type, virtual system version, processor specifiers, developer parameters, domain flags, security information, binding information, administrative flags, and profile information associated with the operating environment of the virtual subsystem.

17. The system of claim 16, wherein the parameters are associated with an identifier to enable selection the native executable.

18. The system of claim 16, wherein the developer parameters describe at least one of debug options, compiler switch settings and information relating to preferences of a user.

19. A computer-readable medium having computer-executable components for executing the system of claim 1.

20. A computer implemented method to generate runtime code, comprising the following computer executable acts:

determining a first code image associated with a possible runtime environment;

executing the first code image in an unmodified form in the runtime environment; and

generating runtime feedback associated with the first code image and a particular user to adjust a subsequent code image according to the runtime environment, the feedback includes at least a set of information to create a code image according to the particular user.

21. The method of claim 20, further comprising,

generating a specialized executable from the subsequent code image.

22. The method of claim 21, further comprising,

storing the specialized executable in an image repository.

23. The method of claim 21, further comprising,

processing a generic intermediate language image utilizing standard compilation techniques.

24. The method of claim 23, further comprising,

logging operating environment information during processing of the generic image.

25. The method of claim 23, further comprising,

building the specialized executable from the environment information.

26. The method of claim 25, further comprising,

selecting the specialized executable by matching a current environment setting with the logged environment information.

27. The method of claim 20, further comprising at least one of:

loading executables with the first code image;

organizing data fields and methods within the first code image; and

analyzing intermediate language formats to generate native platform code.

28. A system to generate runtime code, comprising:

means for determining a specialized code image associated with a possible runtime environment;

means for executing the specialized code image in an unmodified form in the runtime environment; and

means for generating runtime feedback associated with the specialized code image and a particular user to adjust a subsequent specialized code image according to the runtime environment, the feedback includes at least a set of information to create a specialized code image according to the particular user.

29. The system of claim 28, further comprising,

means for generating a specialized executable from the subsequent code image.

30. A computer implemented signal transmitted between at least two processes executing on one or more computers facilitating generation of specialized executables, comprising:

a signal for communicating between one or more components of a virtual system, the virtual system processing a generic code image and logging information relating to an operating environment and a particular user of the virtual system *via* the signal, the logged information

includes at least a set of information to create a specialized native executable according to the particular user;

wherein the logged information is employed as feedback across the signal to generate a specialized native executable if the generic code image is determined incompatible with the operating environment of the virtual system, the specialized native executable is utilized to provide improved performance of the virtual system.

31. The signal of claim 31, wherein the signal is communicated over at least one of a network system and a wireless system.

32. A computer implemented means having stored thereon a data structure employed to create an executable image, comprising:

a first data field having parameters relating to at least one of an operating system version, a processor type, a virtual system version, and a processor specifier;

a second data field having at least one of a developer parameter, a domain flag, a security information field, and a binding information field; and

a third data field having at least a set of information associated with a particular user that is logged during execution of a virtual system to create an executable image according to the particular user of the virtual system.

33. A computer implemented system, comprising the following computer executable components:

an execution engine that processes an Intermediate Language (IL) image, the execution engine generating operating environment data associated with a particular user while processing the IL image;

a specialized executable image generated at least in part from the operating environment data, the operating environment data includes at least a set of information to create a specialized executable image according to the particular user, the specialized executable image stored in a repository of one or more other specialized executable images; and

wherein the execution engine selects at least one specialized executable image from the repository if the at least one specialized image matches present operating environment data.

IX. Evidence Appendix (37 C.F.R. §41.37(c)(1)(ix))

None.

X. Related Proceedings Appendix (37 C.F.R. §41.37(c)(1)(x))

None.